REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Washington Headquarters Service, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington, DC 20503.

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
MARCH 2010	Conference Paper Preprint	June 2008 – March 2010

4. TITLE AND SUBTITLE

IANetServ: Design and Implementation of Robust An Auto-Configurable Network Services for Airborne Network (PREPRINT)

5a. CONTRACT NUMBER
FA8750-08-C-0133

5b. GRANT NUMBER
N/A

5c. PROGRAM ELEMENT NUMBER
65502F

6. AUTHOR(S)

Kyung J. Kwak, Julia Deng, Justin Yackoski, Harry Bullen, Jason Li, Tony McAuley, and Subir Das

5d. PROJECT NUMBER
08SB

5e. TASK NUMBER
IR

5f. WORK UNIT NUMBER
64

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Intelligent Automation, Inc.
15400 Calhoun Drive, Suite 400
Rockville, MD 20855

Telcordia Technologies, Inc.
One Telcordia Drive
Piscataway, NJ 08854-4551

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

AFRL/RIGC
525 Brooks Road
Rome NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)
N/A

11. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFRL-RI-RS-TP-2010-26

12. DISTRIBUTION AVAILABILITY STATEMENT
Approved For Public Release; Distribution Unlimited. PA #: 88ABW-2010-1754
Date Cleared: 31-March-2010

13. SUPPLEMENTARY NOTES
This work, resulting in whole or in part from Department of the Air Force contract number FA8750-08-C-0133, has been submitted for publication in the MILCOM 2010 proceedings. If this work is published, IEEE may assert copyright. The U.S. Government has for itself and others acting on its behalf an unlimited, paid-up, nonexclusive, irrevocable worldwide license to use, modify, reproduce, release,perform, display, or disclose the work by or on behalf of the Government.All other rights are reserved by the copyright owner.

14. ABSTRACT
The next-generation airborne networks (ANs) will be all IP-based to provide seamless connectivity and reach ability among network hosts including diverse aircrafts and unmanned air vehicles. Due to the dynamic nature and bandwidth/capacity limited channel condition of airborne network, providing dynamic yet efficient network service always faces many challenges. In this paper, we summarize the IANetServ architecture, which provides robust and auto-configurable network service framework suitable for performance of the IANetServ architecture. Amongst many different types of network service, we limit our presentation of the IANetService architecture to the address configuration and name resolution.

15. SUBJECT TERMS
Network Services, Airborne Networks, DHCP, DNS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Bradley J. Harnish
U	U	U	UU	8	19b. TELEPHONE NUMBER (*Include area code*) N/A

IANetServ: Design and Implementation of Robust and Auto-configurable Network Service for Airborne Network

Kyung J. Kwak, Julia Deng, Justin Yackoski, Harry Bullen, Jason Li
Intelligent Automation, Inc.
Rockville, MD

Tony McAuley, Subir Das
Telcordia Technologies, Inc.
Piscataway, NJ

ABSTRACT

The next-generation airborne networks (ANs) will be all IP-based to provide seamless connectivity and reachability among network hosts including diverse aircrafts and unmanned air vehicles. Due to the dynamic nature and bandwidth/capacity limited channel condition of airborne network, providing dynamic yet efficient network service always faces many challenges. In this paper, we summarize the IANetServ architecture, which provides robust and auto-configurable network service framework suitable for airborne network environments. We also present our implementation effort into simulation and hardware to evaluate the performance of the IANetServ architecture. Amongst many different types of network service, we limit our presentation of the IANetServ architecture to the address configuration and name resolution.

1. INTRODUCTION

Airborne Networks (ANs) have recently received enormous attention as a new emerging all IP-based network technology. It has been envisioned as an infrastructure to provide communication services to a set of heterogeneous mobile airborne platforms by interconnecting airborne, terrestrial and space network together [2]. ANs are still in its early stage of development and commonly believed to have some flavor of dynamic mobile ad hoc networks (MANETs), i.e., no fixed infrastructure will be supported and nodes join or leave the network frequently without planning. Given the dynamically changing topology, limited network bandwidth, and self-organizing nature of domains due to high node mobility, designing an appropriate network service framework also confront these challenges.

Some initial research efforts [2]-[5] have been made to address these challenging issues. An auto-configuration approach for network services in the AN environments [2] was proposed by using randomly generated 57bit subnet ID, multicast DNS for name service. This approach provides a low possibility of address collisions for relatively large network (i.e., 1000 airborne nodes). However, there is always possibility of address collision. In addition, this approach is not compatible with existing standard implementations and lack of security considerations.

In [4], an Integrated robust and Auto-configurable Network Service (IANetServ) framework was initially designed in order to provide robust and auto-configurable network service support in ANs. For network configuration, an automatic conflict-free address assignment was developed with built-in mechanisms to further support network dynamics, such as domain splitting and merging. For name resolution service, the concept of Logical Name Service (LNS) was introduced to provide robustness and efficiency. We also extended the IANetServ framework to be diretly applied to the JANSS (Joint Airborne Network Service Suite) architecture [6][7]. In addition, the IANetServ framework was enhanced by adding security boundary to make it fit into the security enhanced airborne platform for information assurance.

As a continuous effort, we implement the extended IANetServ framework on our in-house agent-based network simulator and wireless routers. In this paper, we present our implementations of the extended IANetServ framework in detail. It is noted that amongst diverse network services, we focus more on the address service and name service.

The rest of the paper is organized as follows. Section 2 reviews the JANSS design document for better understandings of the JANSS architecture and the extended IANetServ framework. Section 3 describes our implementation on our in-house java simulator and Linksys wireless router platform. Section 4 concludes IANetServ design and implementation followed by a short acknowledgement.

2. NETWORK SERVICE ARCHITECTURE

In this section, we briefly summarize the (1) JANSS, which defines overall features and functions for Joint Airborne Network (JAN) and (2) IANetServ framework, which enables dynamic and autonomous network service among heterogeneous airborne platforms. More detail description of each IANetServ components can be found at [3] and [4].

2.1 JANSS architecture

The JANSS design document draws the whole picture of integrated features, concept, standard and functions which enables each airborne platform participate in the future all

IP-based AN. The JANSS will allow air and maritime platforms to be interoperable among themselves, and with other edge tactical networks and Intelligence, Surveillance and Reconnaissance (ISR) networks through transparent IP connectivity. The required functional blocks for airborne platform include name service, address service, service discovery, mobility management, security service, and network management. It also defines a simplified airborne platform as in Figure 1. Each airborne platform supports and manages multiple local area networks (LANs) through AN router which functions as a gateway and maintains several RF links via IP supported radios. Each IP radio also has Mobile Network Router which provides routing capability within each RF link.

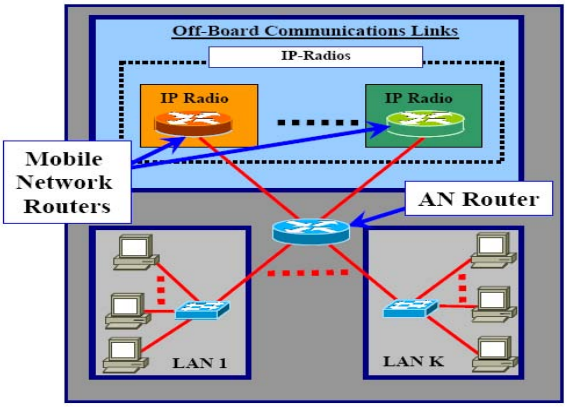


Figure 1: Airborne Network Platform in JANSS [6]

The current JANSS architecture assumes the airborne network operates at single security level (i.e., data protection is performed via link encryption only). However, in reality, the access to information and services are based on the different sensitivity level classifications. These Multiple Independent Levels of Security have specific policies, procedures and requirements. A user must not be able to read information at a security level higher than its own, nor be able to write information to a lower classification level. In fact, information must be mutually invisible among levels/classifications/compartments. As it is very inefficient for each security level to have its own physical network, each classification level will transit through an independent Secure Virtual Private Networks (VPNs) over a shared backbone.

2.2 IANetServ framework

The IANetServ framework mainly consists of two network services: address configuration, and name resolution.

For address configuration service, the ADCA (Adaptive Domain Configuration Agent), DAP (Domain Announcement Protocol), DHCP (Dynamic Host Configuration Protocol), and DRCP (Dynamic and Rapid Configuration Protocol) were designed to provide automatic conflict-free address assignment under bandwidth constraint environment. Similar to the legacy Dynamic Host Configuration Protocol (DHCP), the DRCP provides IP address configuration within one-hop proximity. The DCDP distributes IP address pool to one hop neighboring nodes by splitting its own address pool to ensure conflict-free configuration. By combining both DRCP and DCDP processes together, a subset of nodes in the large network carry a chunk of IP address space and assign the IP addresses to other non-configured nodes. In doing so, they perform as a distributed and independent address server. The DAP is a domain maintenance protocol which handle network splitting and merging by using periodic beacons. The ADCA controls configuration process by monitoring what information has been distributed through the DCDP and DRCP processes in each domain.

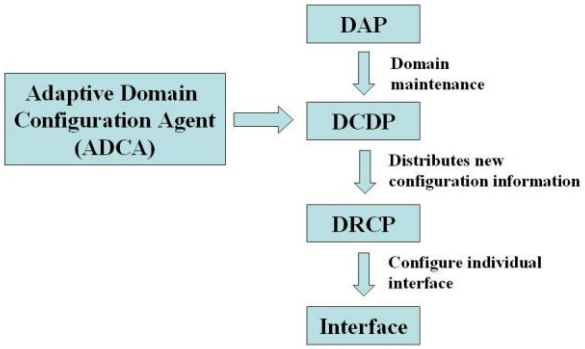


Figure 2: Address auto-configuration mechanism

For name resolution service, we proposed the Logical Name Service (LNS), which is an extension of legacy Domain Name System (DNS). The fundamental philosophy of LNS is based on the ideas of mobile IP and the work in [10][11]a. A LNS server maintains three different mapping tables, i.e., Topological, Logical, and Home tables. The topological table contains name to IP address mapping within its own domain. The logical table maps a group of logical names to the IP address of the logical home servers. The home table maps a particular logical name to the IP address of the server that is currently responsible for the specific entity.

While designing the LNS mechanism, we put special emphasis on handling both node and name server mobility to address the high dynamics in airborne environments. Such goal could be achieved by adding dynamic server configurations and autonomous linkages to enhance its robustness and efficiency. In each domain, a name server is dynamically elected to provide name resolution service in its own domain. A backup name server might also be selected when necessary.

2.3 IANetServ framework extension

We extended the IANetServ framework to be compatible with JANSS architecture. Besides of address configuration and name resolution service, the IANetServ framework considers the various levels of security aspect at each airborne platform. In our extended design, the NSA-endorsed Inline Network Encryptors (INE) or High Assurance Internet Protocol Encryptor (HAIPE) device [10][11] divides airborne platform into the untrusted black network and trusted red network. The key issue of extending the IANetServ framework is where the different IANetServ components should reside in the JANSS architecture to provide efficient yet secure network service. Figure 3 shows our proposed extended design, where a set of name servers, ADCA agents and address configuration protocols provide configuration and naming services on both red and black network.

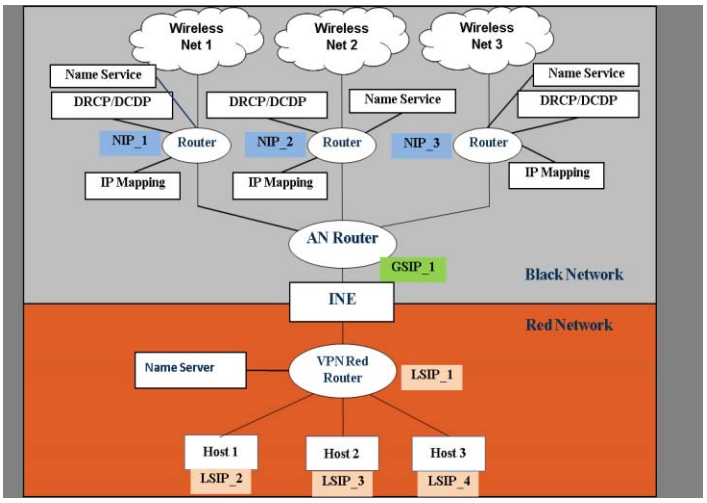


Figure 3: IANetServ framework in JANSS architecture

The ADCA, DAP, DRCP, and DCDP stay at the black network. The ADCA handles IP address management including maintaining address pools, address lease/collection, domain merging/splitting and updating configuration policy for reliable and autonomous configuration service. In particular, the DAP/DRCP/DCDP protocols will be placed on each IP radios which belong to different tactical network. Each air platform is associated with multiple IP addresses, static IP address (SIP) and dynamic network IP address (NIP). There is a global static IP address (GSIP_0 in the figure), associated with the AN Router on each air platform. The GSIP works as a global ID for the air platform.

Our address configuration protocols also can be deployed in the red network. However, note that the hosts in red network are usually wired connected and their addresses are static. In this case, standard auto-configuration protocol such as DHCP can also be used. There is a set of local static IP addresses, associated with Red Router and local hosts (LSIP_1, LSIP_2, LSIP_3, LSIP_4). The Red Router works as a gateway router for the internal LAN on each air platform. Each mobile router in the wireless net is associated with a dynamically generated IP address, NIP.

To ensure a successful communication between any two hosts (computers) located in any two air platforms through each type of wireless radio connection, each wireless communication radio will have a corresponding *IP Mapping* component to map the NIP address with the corresponding GSIP address. To maintain the network topology and connectivity information, each airborne platform need periodically exchange/update network information with others during the network operation time. In particular, the associated NIP of each airborne platform is updated in the *IP mapping* table.

The LNS protocol also resides on the black network to provide naming service. To handle high node dynamics in a intermittent and bandwidth limited environments, table synchronization among LNS servers is critical. Our implementation and thorough evaluation confirms that LNS protocol handles node dynamics reliably and efficiently. This is exactly why LNS is proposed as part of name resolution service in IANetServ. As all user applications are on the red side, name resolution service is also required on the red network. While placing a name server on the red network, there are two contradictory goals. To ensure robust and autonomous operation, it is required to have redundant name servers. In contrast, it is also desirable to keep fewer name servers to limit the use of scarce network bandwidth, particularly when there are a significant number of updates. To meet requirements of airborne network, we believe one name server per red side of platform is a good solution.

3. IANetServ Implementation

For realistic feasibility study and thorough performance evaluation, we implemented the IANetServ architecture in IAI's Java based discrete network simulator NetSim and Linksys WRT54G router. More detail description of each implementation follows.

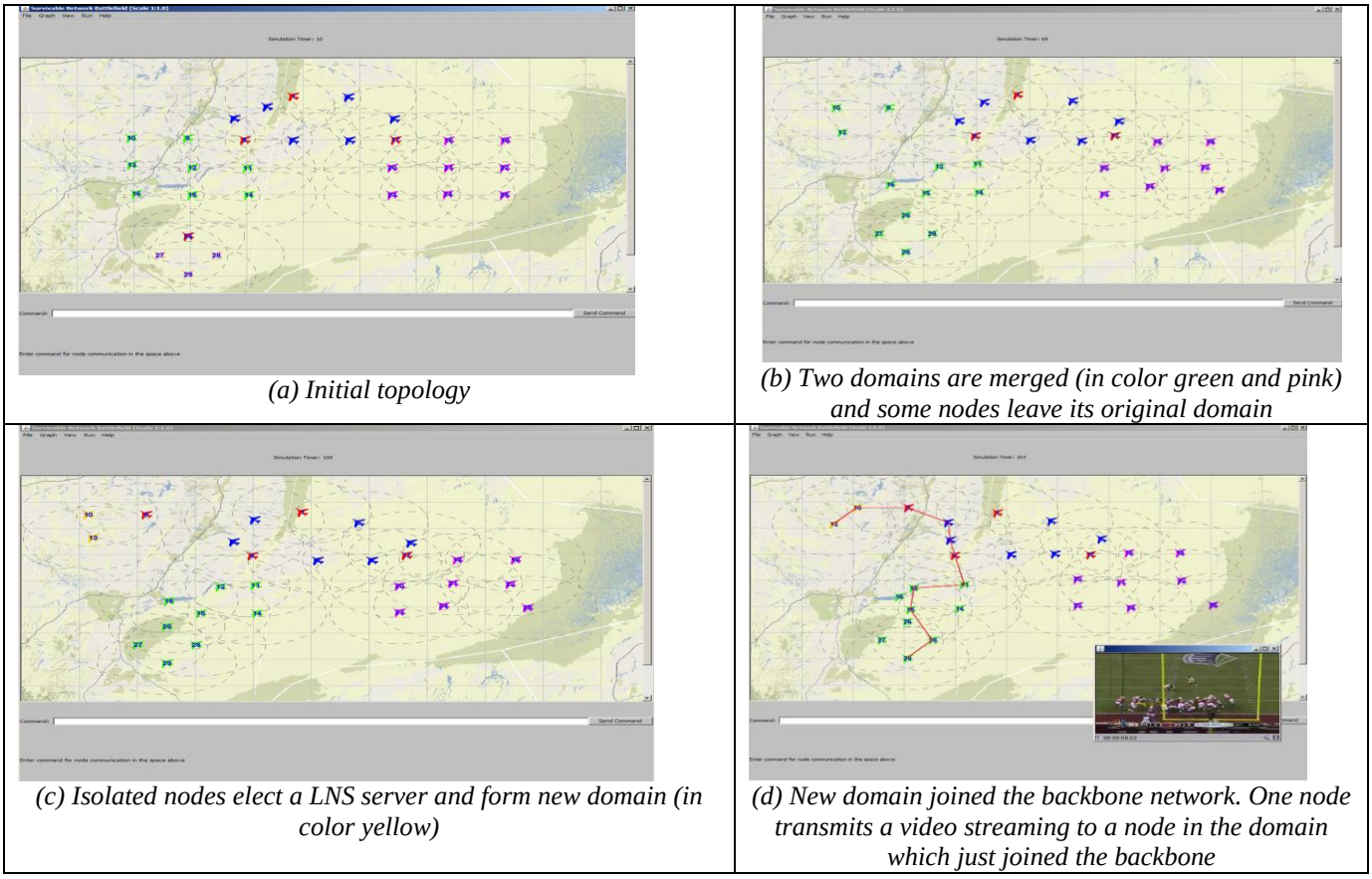
3.1 NetSim Simulation

The NetSim simulator is specially developed for simulation and evaluation of advanced distributed algorithms for ad hoc mobile networks. It is based on over a decade of development effort of software agent-based infrastructure Cybele (http://products.i-a-i.com/), which is a Java-only layer that can be customized for the desired platform. It is a runtime environment built on the top of the Java 2 platform for control and execution of agents. The agent execution is, however, event-driven and controlled by Cybele© rather than by JVM. A common basic network node structure

provides the basis from which the network nodes for all protocols are derived. This basic network agent component is responsible for interacting with the simulation environment and for providing the software under test with the same inputs as it would experience if it were implemented in a real device.

The NetSim simulator is able to explore distributed computing capabilities for evaluation of scalability of protocols, without traffic aggregation or pre-calculation steps, which allows for human-in-the-loop experimentation. In this architecture the nodes to be simulated are implemented using autonomous agents. Each agent/node generates, receives and processes packets independently, as it would on a real network. It is important to note that the code executing on each agent is no different from the code that would actually execute on the real system, except for the way in which they communicate with the environment.

In our IANetServ architecture implementation in NetSim, each node is defined as an individual agent and its configuration is set by scenario file and configuration file. A scenario file specifies node movement (e.g., speed, path shape, relay point, etc) and enable/disable application access (e.g., data or real time traffic transfer). On the other hand, a configuration file defines initial node position and node specific information, e.g., host name, domain name, priority, initial IP pool size, etc. All detail functions of IANetServ are implemented, e.g., DRCP, DCDP, DAP, and LNS activities, etc. Each node selectively calls activities when necessary. We assume several tactical edge network is interconnected through a backbone network.



(a) Initial topology

(b) Two domains are merged (in color green and pink) and some nodes leave its original domain

(c) Isolated nodes elect a LNS server and form new domain (in color yellow)

(d) New domain joined the backbone network. One node transmits a video streaming to a node in the domain which just joined the backbone

Figure 4: Domain merging and splitting

The Border Gateway Protocol (BGP) is the routing protocol used in backbone network to properly deliver LNS queries and responses among LNS. To achieve this goal, BGP and network service protocol suite were integrated with each other such that any BGP node can configure its IP address and get proper naming service through LNS server. Within each edge network, the Ad hoc On-Demand Distance Vector Routing (AODV) is used to transport data packets as well as network service protocol packets. The mobility pattern of backbone nodes mimics the Caspian Sea scenario (i.e., moving path forms a long eclipse), and the nodes in edge network follow some customized the mobility to demonstrate different network scenarios.

Here we present two simulation scenarios. The first scenario tests a case where one node joins and leaves a domain. Although other nodes also fairly move, we call

that node the mobile node since its speed is much higher than others. Before the mobile node switches its domain, another node in the same domain transmits a video streaming to the mobile node. While the mobile node enters a new domain and reconfigures its address, a video transmission will be halted since the old IP address is no longer valid. Once a new IP address is assigned and all LNS tables are updated, the video transmission will be resumed. Note that a source node will send a LNS query again to its LNS server after it recognizes the mobile node's domain change and IP address reconfiguration of the mobile node accordingly. Since all LNS servers are part of backbone network, all LNS queries and updates will be routed through BGP. Routing path is marked in red line and a popup window displays the video stream that has received.

Figure 4 shows snapshots of the second scenario, where several domain merging and splitting are handled. In Figure 4, nodes in blue and red are backbone nodes which interconnect with two tactical edge networks in color green and purple, and third edge network (in color yellow) is isolated. As the domain in color yellow heads north, it receives a LNS broadcasting with higher domain priority (in color green). Upon receiving a LNS broadcasting, nodes merge into the new domain and reconfigure their IP address through DRCP/DCDP processes. Simultaneously, three nodes in green will leave their domain and form a new domain (see Figure 4(b)).

As nodes are isolated, they cannot listen any LNS broadcasting and realize domain split. Later, new LNS server will be elected within the newly formed domain and it will also assign new IP address to all members (see Figure 4(c)). The three nodes keep moving until they are reconnected through the backbone network, and then a node in bottom domain transmits a video file to a node on top (see Figure 4(d)). In real scenarios, airborne platforms might have satellite link to prepare for the case when backbone connectivity is lost. However, we assume inter-domain communication always go through the backbone network. Through the above two testing scenarios, we observed that the IANetServ architecture successfully handles domain merging/splitting and IP address assignment to dynamic nodes and resolves name in robust and efficient fashion. This simulation also confirms that LNS tables at each LNS server are efficiently and reliably synchronized.

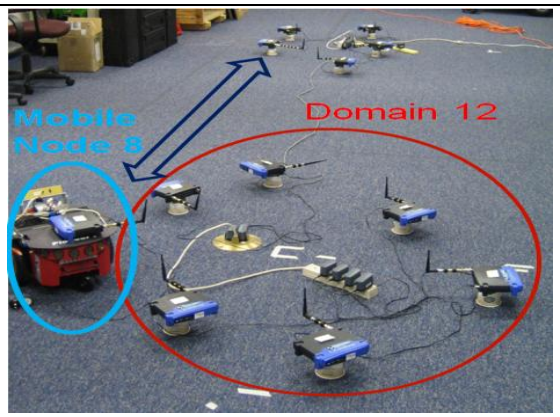
3.2 Router implementation

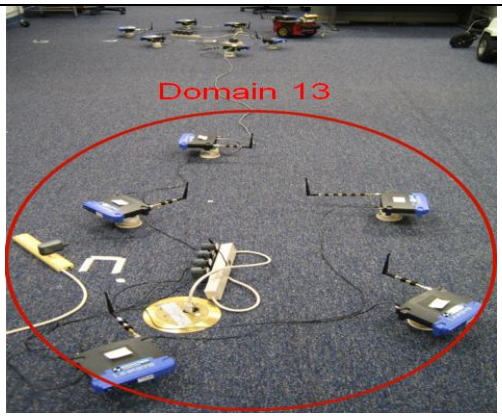
To further test the performance of the IANetServ framework, we implemented it on the Linksys WRT54G routers. The router is selected since it is notable for being the first consumer-level network device that had its firmware source code released to satisfy the obligations of the GNU. This allows programmers to modify the firmware to change or add new functionality. Its CPU speed is 240 MHz and the size of RAM and Flash is 8 MB and 2MB respectively. The OpenWrt [12] firmware is used in this research. To control network topology as we wanted and minimize interference among routers, we applied 50dB RF signal attenuator to the front of antenna. Note that communication range has shrunk from 150 feet to about 10 feet.

The DRCP and DCDP protocols are implemented in Linux using a scripting language and a custom program written in C to perform the actual sending of packets via unicast and/or broadcast. All protocols messages are sent via UDP as human-readable ASCII text. Both DRCP and DCDP are implemented as a pair of actor and listener processes, with the actor process initiating actions on behalf of the current node and the listener process handling packets received on the DRCP and DCDP ports. The LNS implementation consists of two components. First, nodes use the LNS mechanism to look up another node's IP address on top of the legacy DNS. Each LNS node runs an unmodified DNS server where any changes of the IP address to host name mapping triggers DNS server configuration update and reload. Second, to perform updates between multiple domains, LNS nodes periodically exchange LNS_UPDATE messages, advertising IP addresses assigned within their domain using the same format as in the DRCP_RENEW with the node ID, assigned IP, and time elapsed. By comparing the time elapsed in the LNS_UPDATE message from another domain with the LNS node's record of any IPs assigned within its own domain, each LNS node can determine which IP assignment is most recent and use this assignment in response to DNS queries. The LNS_UPDATE messages can be easily incorporated into BGP or other routing data, however when few domains are present this may save minimal overhead and result in additional delay updating the IP and ID mappings.

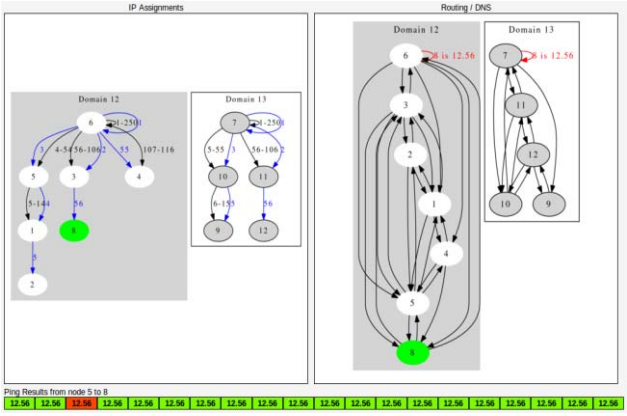
Figure 5(a) shows our test setup where a mobile node switches its domain back and forth. Each domain consists of 5~6 routers which form one or two hop neighbors with each other. Each router competes with members in a domain to be elected as a LNS server. We use uptime of each router as the mean of priority (i.e., longer uptime has higher priority). One router (node 8) was put on the robot which travels between two domains (domain 12 and 13). For the debugging and demonstration purpose, we also developed a Graphic User Interface (GUI) as shown in Figure 5(b). Left panel shows the progress of IP pool and address assignment. Arrows indicates which nodes is assignee and assigner. Numbers indicate the last eight bits of IP address. Note that each domain has its own IP prefix.

Right panel shows connectivity and routing path. We used the Optimized Link State Routing (OLSR) protocol in Quagga package [13] for intra-domain routing. To emulate backbone connectivity, we used company Ethernet for LNS nodes to exchange and update their LNS tables. To validate and evaluate the performance of IANetServ implementation, we set a node in domain 12 periodically ping the mobile node throughout the test. The green and red squares at the bottom of GUI show whether ping message was successful or not. In GUI display, green means ping message goes through.

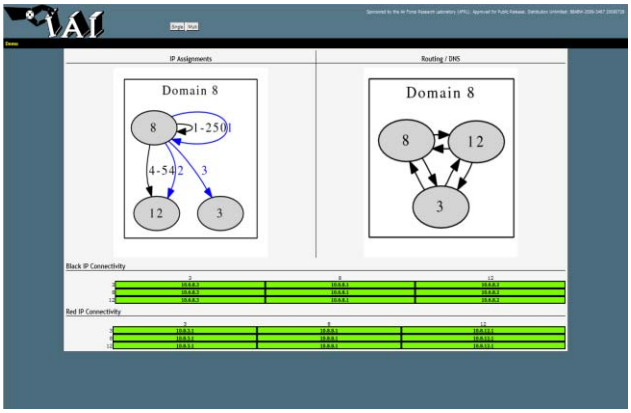




(a) Test setup with 3 domains and one mobile router



(b) GUI display



(c) GUI display with red/black network connectivity

Figure 5: IANetServ router implementation and GUI design

To implement emulation of red/black network separation, we assign each router a unique host name ID.iai, which is updated and served by the LNSs. This corresponds to GSIP and it is map the router's black IP address. We also assigned a unique red IP address, since red IP address does not change frequently. The Black IP address was the only one that dynamically assigned and updated at each LNS server. We use bridge control to interconnect emulated red and black network. In our GUI as in Figure 5(c), the connectivity between multiple black IP addresses is represented dynamically by asking routers to ping the ID.iai (or GSIP) of other routers. This process requires mapping between GSIP and black IP address and name resolution at each LNS server. To emulate the VPN tunnel between any pair of red network entities, Vtun [14] is used to setup VPN tunnels. Vtun is a tunneling program with a client/server model. It is flexible in supporting many types of tunneling and allowing for traffic to be compressed and encrypted. The bottom table of GUI shows reachability of any pair of red host through bridge control and Vtun.

4. CONCLUSIONS

The topology of airborne networks continuously changes and channel bandwidth is limited, thus autonomous and reliable yet efficient network service architecture is highly desired. In this paper, we summarized our effort on designing an Integrated robust and Auto-configurable Network Service (IANetServ) architecture to handle the network dynamics and bandwidth constraints over airborne networks. We also presented our implementation on agent based Java simulator and Linksys routers. Our evaluation

using simulation and hardware implementation confirmed the effectiveness and robustness of IANetServ architecture. While existing works on network service dedicated to airborne networks are either limited to few aspect of network service or based on the approached in wired network, the IANetServ architecture is the first complete solution for robust, scalable, and autonomous network service in airborne network environments. Our on-going effort is to finalize our hardware implementation and polish our design.

ACKNOWLEDGEMENT

We would like to express our appreciation to AFRL for funding this research with contract number FA8750-08-C-0133. We would also like to give our thanks to our program managers Mr. Bradley Harnish and many research associates from AFRL and MITRE for their valuable suggestions and advices during this research.

REFERENCE

[1] Daryl Mayer, "Airborne Network to link sensors, shooters, decision makers," http://integrator.hanscom.af.mil/2005/February/02242005-02.htm

[2] USAF Airborne Network Special Interest Group, "Airborne Network Architecture" version 1.1, October 2004,

[3] L. Veytser, T. Shepard, J. Colley and V. Mehta, "Autoconfiguration of network services in Airborne wireless networks," IEEE MILCOM 2005.

[4] H. Deng, J. Li, R. Xu, T. McAuley, S. Das, D. Agrawal, "An Integrated Robust and Auto-Configurable Network Service Protocol Suite for Airborne Networks" IEEE MILCOM 2008.

[5] Steven V. Pizzi, "A Routing Architecture for the Airborne Network," MITRE 2007.

[6] Joint Airborne Network Services Suite, US Air Force standards definition document, Nov. 2006.

[7] R. Trafton, S. V. Pizzi, "The Joint Airborne Network Services Suite," IEEE MILCOM 2006.

[8] G. Nakamoto, "Scalable HAIPE Discovery," In Visualizing Network Information Meeting 2006.

[9] M. Mirhakkak, P. Ta, G. Comparetto, V. Fineberg, "Modeling and Simulation of Haipe," MILCOM, pp.1-7, MILCOM 2006.

[10] R. Morera and A. McAuley, "Adapting DNS to dynamic ad hoc networks," Proceedings of IEEE MILCOM 2005.

[11] A. Misra, S. Das, A. McAuley, and S. K. Das, "Autoconfiguration, Registration, and Mobility Management for Pervasive Computing," IEEE Personal Communications, Volume 8, Issue 4, pp. 24-31, 2001.

[12] OpenWrt, http://openwrt.org/

[13] Quagga package, http://www.quagga.net/

[14] Virtual tunnel, http://vtun.sourceforge.net/